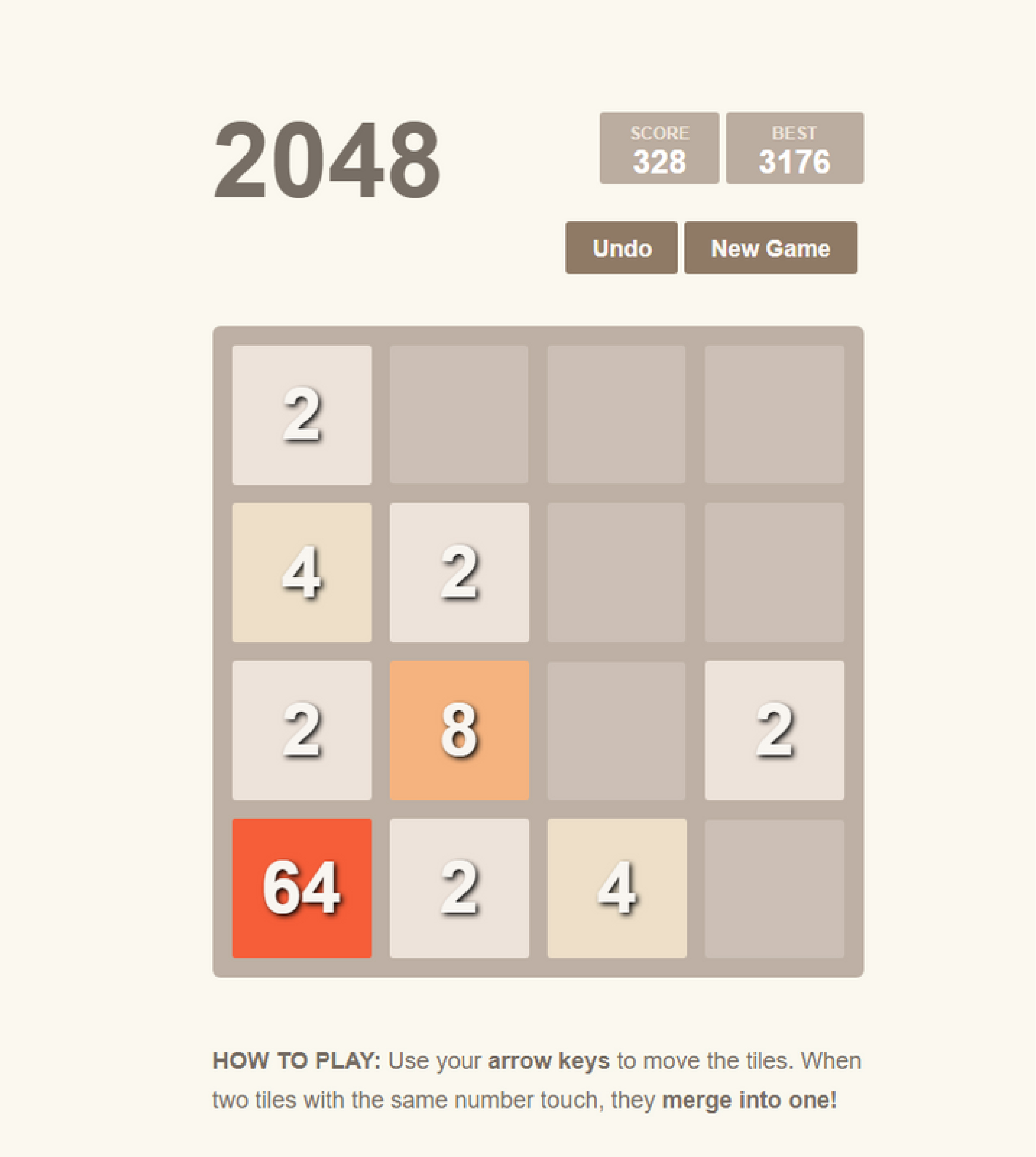
DABA 2048 GAME

www.2048.phamducdat.id.vn

START MENU

What is 2048

2048 is a puzzle game where you combine numbered tiles on a 4x4 grid to reach the 2048 tile. Slide tiles to merge and strategize for high scores.



HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

How To Play

YES

NO

How To Play

- **Game Board:** 4x4 grid with an even number of squares. 2 cells start with 2 or 4, the rest are empty.
- **Moving Tiles:** The player can slide tiles up, down, left, or right. Similar numbers merge, creating a new tile. Empty cells randomly fill with two.
- **Objective:** Reach 2048 tile. Can continue for higher values.
- **Game Over:** No valid moves left (no empty cells or neighboring cells with different values).

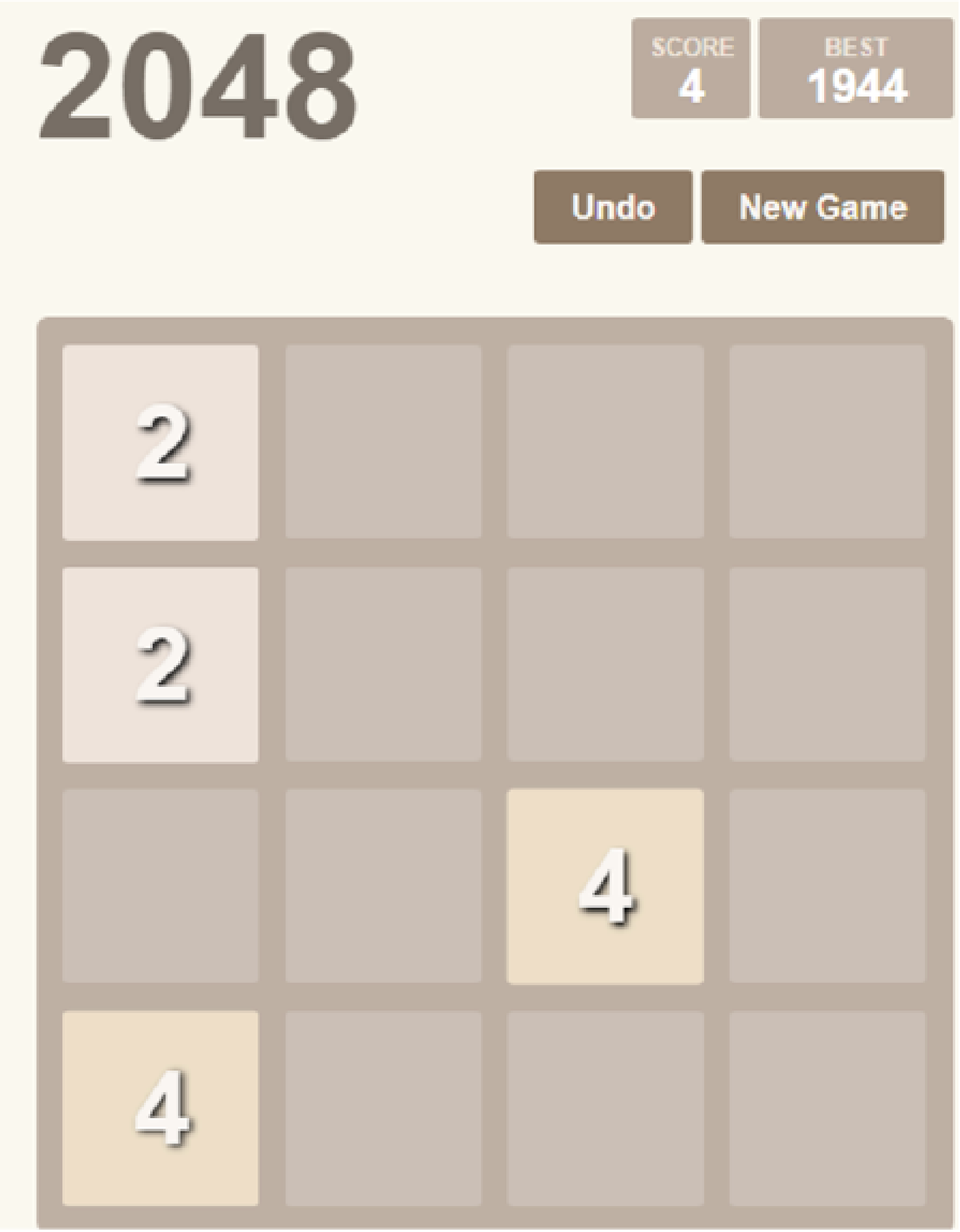
Scoring

YES

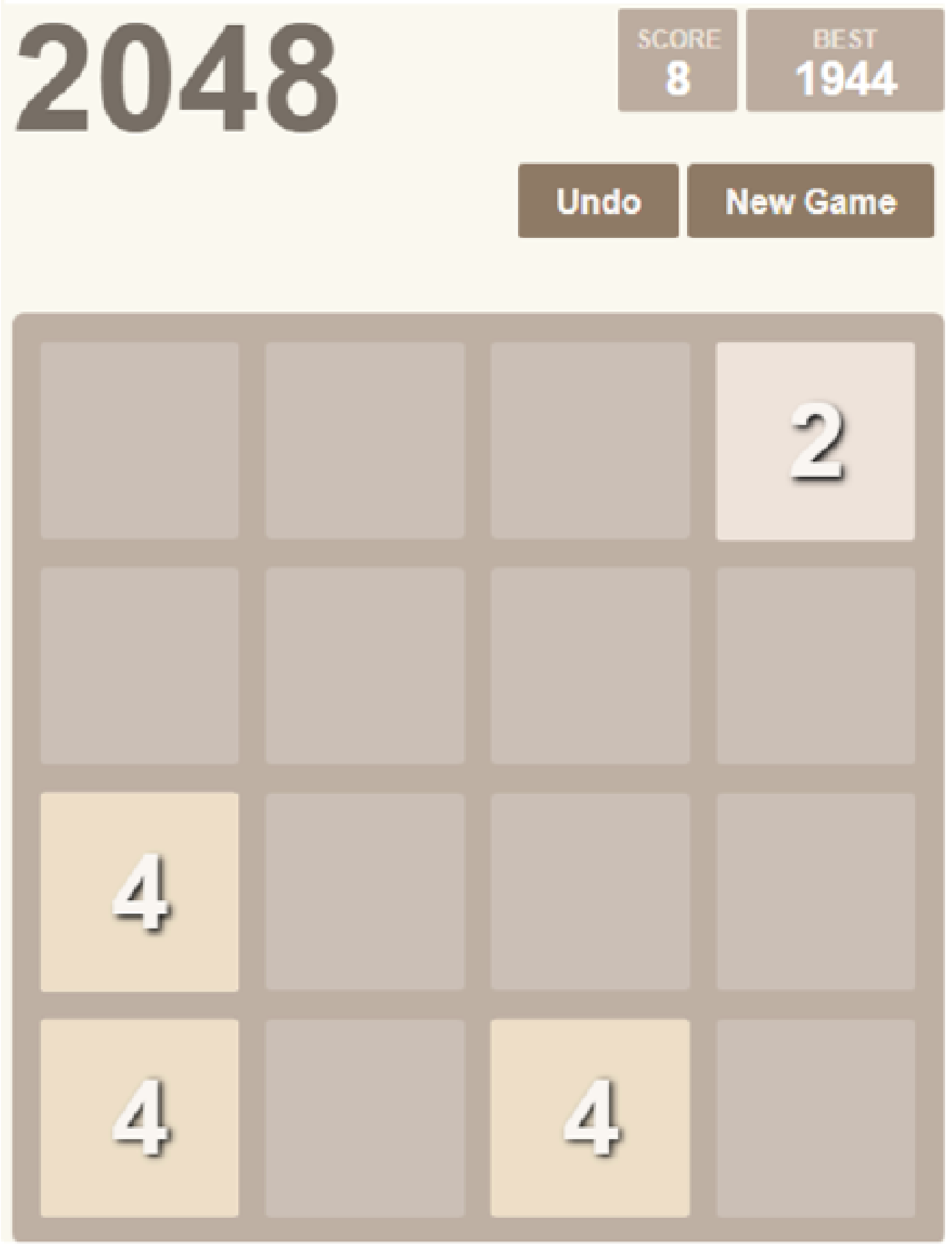
NO

Scoring

- **Scoring:** Matched tiles earn points equal to their combined value.
- Current and highest scores are displayed for player progress.



Before adding score



After adding score

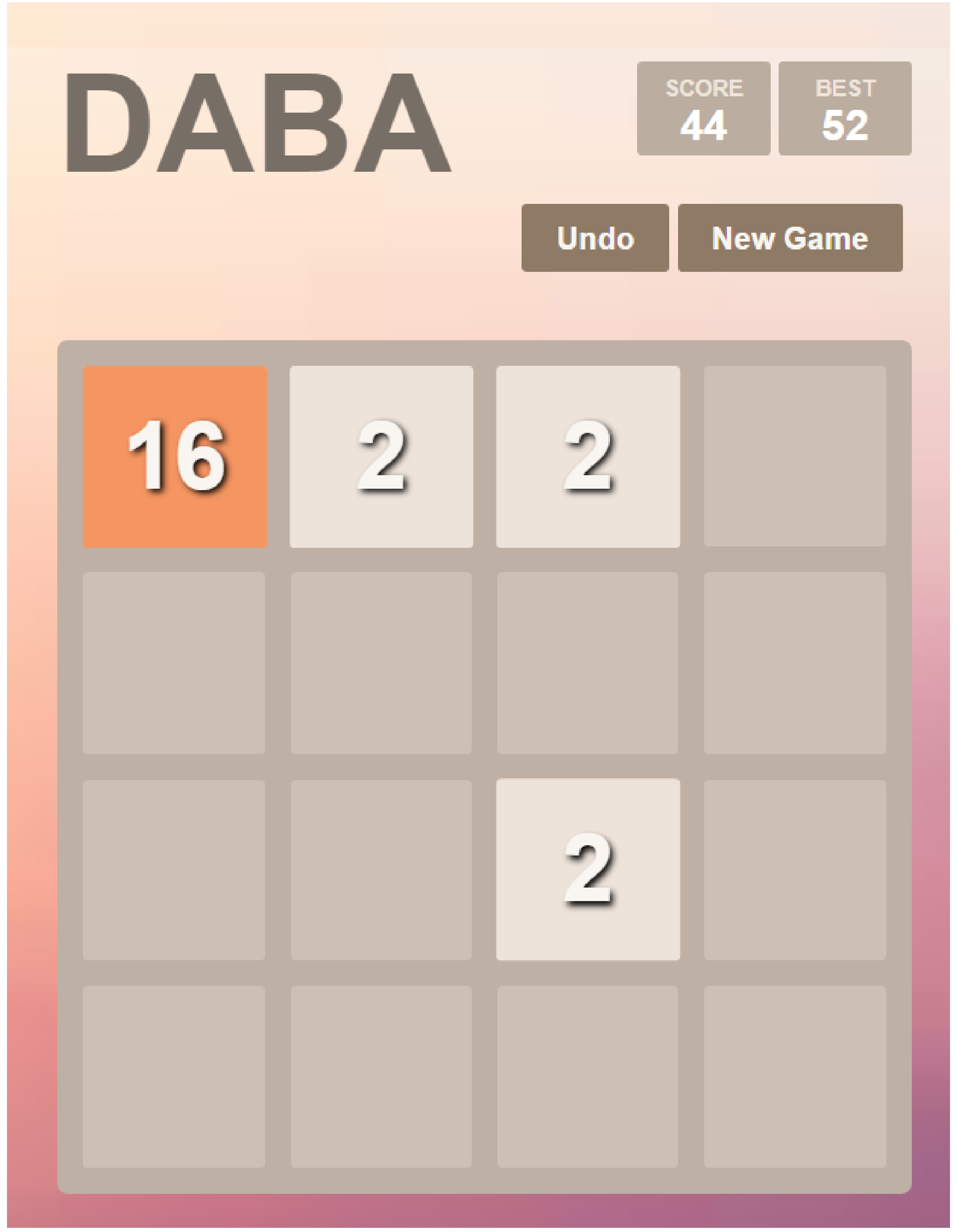
Undo Function

YES

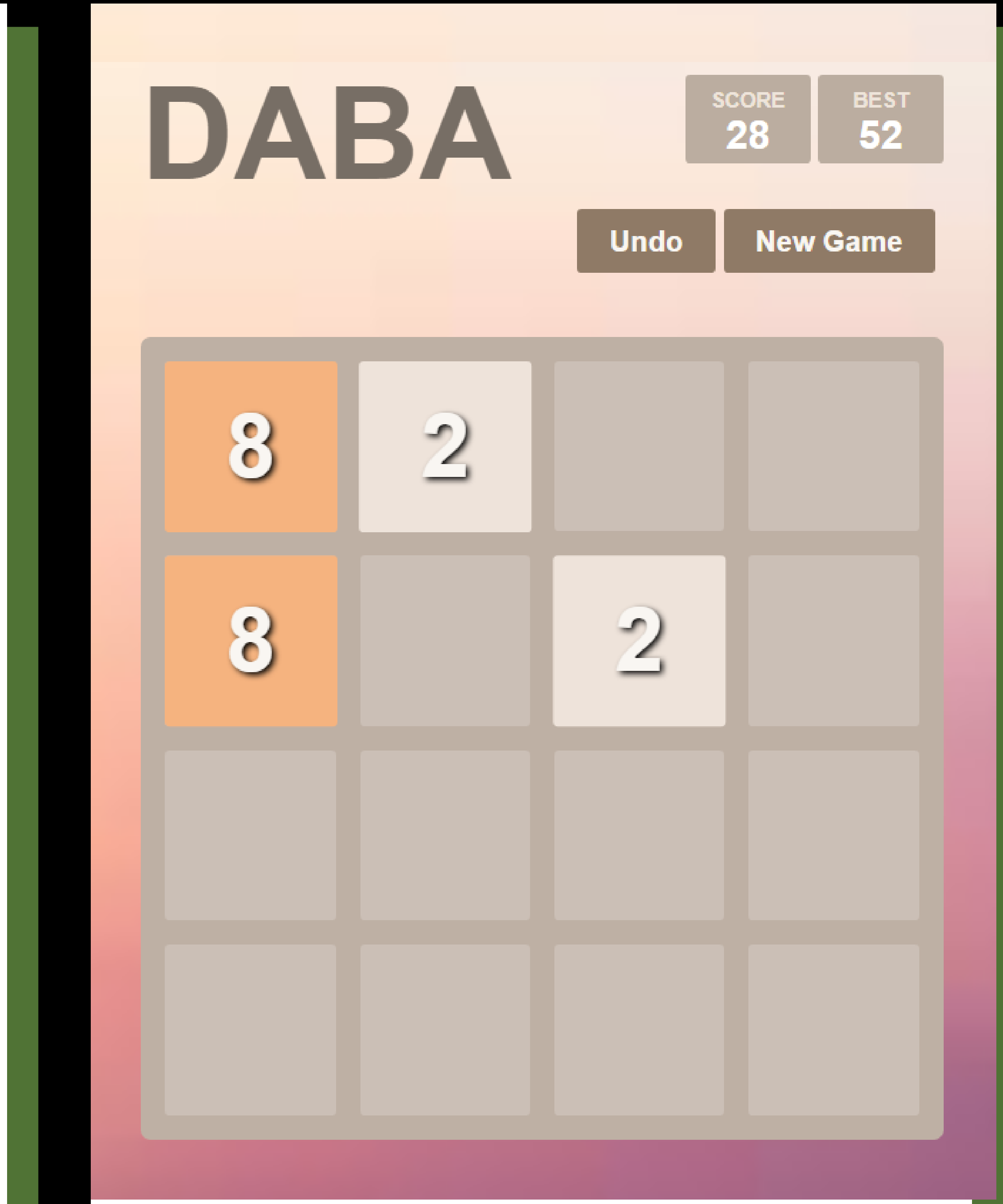
NO

Undo Function

- **Undo Function:** Allows players to reverse their last action.
- **How to Use:** Press the undo button to revert to the previous state and adjust the score accordingly.



Before undo

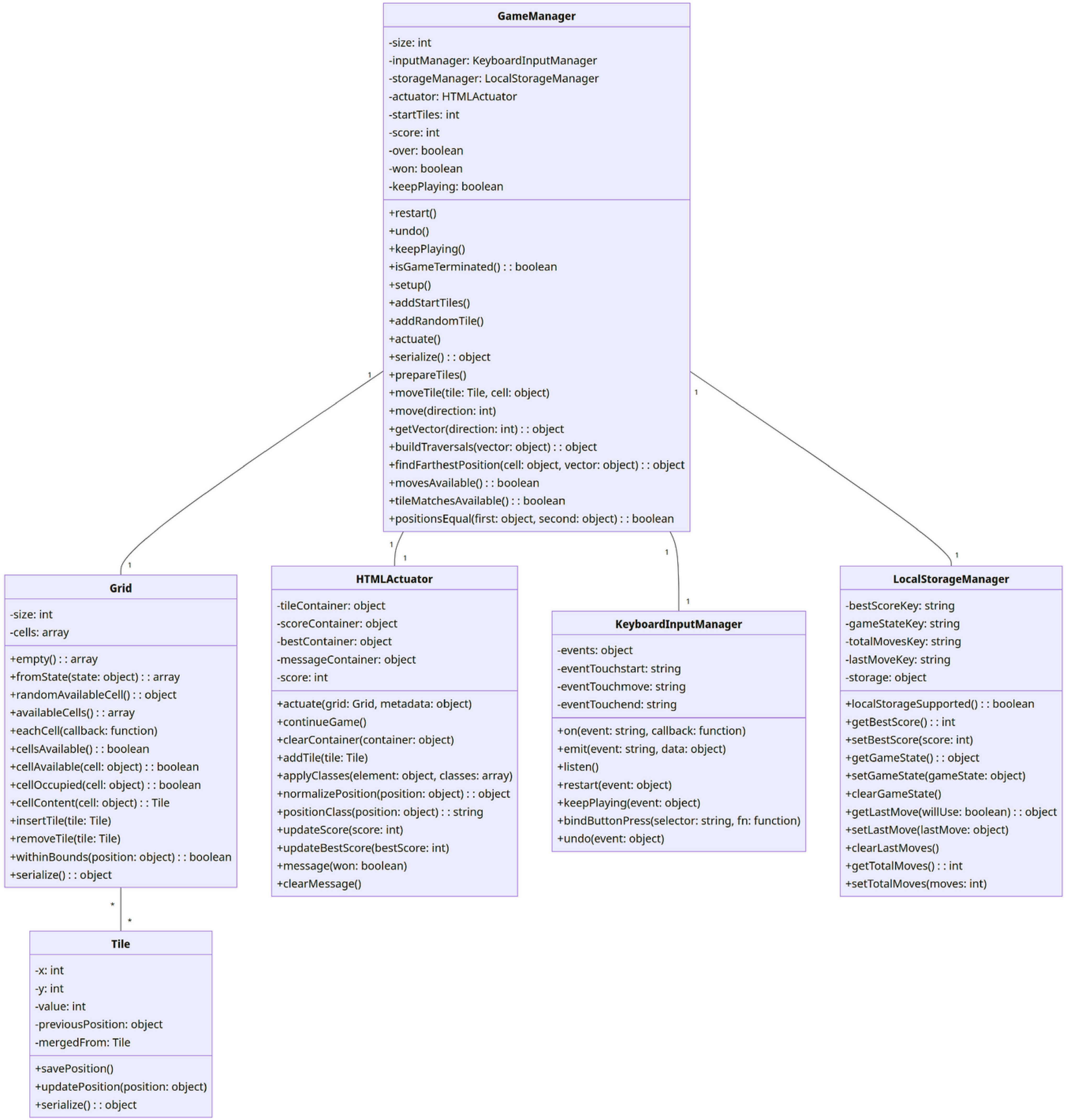


After undo

Class Diagram

YES

NO



GameManager

-size: int
-inputManager: KeyboardInputManager
-storageManager: LocalStorageManager
-actuator: HTMLActuator
-startTiles: int
-score: int
-over: boolean
-won: boolean
-keepPlaying: boolean

+restart()
+undo()
+keepPlaying()
+isGameTerminated() : : boolean
+setup()
+addStartTiles()
+addRandomTile()
+actuate()
+serialize() : : object
+prepareTiles()
+moveTile(tile: Tile, cell: object)
+move(direction: int)
+getVector(direction: int) : : object
+buildTraversals(vector: object) : : object
+findFarthestPosition(cell: object, vector: object) : : object
+movesAvailable() : : boolean
+tileMatchesAvailable() : : boolean
+positionsEqual(first: object, second: object) : : boolean

Grid

-size: int
-cells: array

+empty() : : array
+fromState(state: object) : : array
+randomAvailableCell() : : object
+availableCells() : : array
+eachCell(callback: function)
+cellsAvailable() : : boolean
+cellAvailable(cell: object) : : boolean
+cellOccupied(cell: object) : : boolean
+cellContent(cell: object) : : Tile
+insertTile(tile: Tile)
+removeTile(tile: Tile)
+withinBounds(position: object) : : boolean
+serialize() : : object

HTMLActuator

-tileContainer: object
-scoreContainer: object
-bestContainer: object
-messageContainer: object
-score: int

+actuate(grid: Grid, metadata: object)
+continueGame()
+clearContainer(container: object)
+addTile(tile: Tile)
+applyClasses(element: object, classes: array)
+normalizePosition(position: object) : : object
+positionClass(position: object) : : string
+updateScore(score: int)
+updateBestScore(bestScore: int)
+message(won: boolean)
+clearMessage()

KeyboardInputManager

-events: object
-eventTouchstart: string
-eventTouchmove: string
-eventTouchend: string

+on(event: string, callback: function)
+emit(event: string, data: object)
+listen()
+restart(event: object)
+keepPlaying(event: object)
+bindButtonPress(selector: string, fn: function)
+undo(event: object)

LocalStorageManager

-bestScoreKey: string
-gameStateKey: string
-totalMovesKey: string
-lastMoveKey: string
-storage: object

+localStorageSupported() : : boolean
+getBestScore() : : int
+setBestScore(score: int)
+getGameState() : : object
+setGameState(gameState: object)
+clearGameState()
+getLastMove(willUse: boolean) : : object
+setLastMove(lastMove: object)
+clearLastMoves()
+getTotalMoves() : : int
+setTotalMoves(moves: int)

Tile

-x: int
-y: int
-value: int
-previousPosition: object
-mergedFrom: Tile

+savePosition()
+updatePosition(position: object)
+serialize() : : object

Graphical User Interface

YES

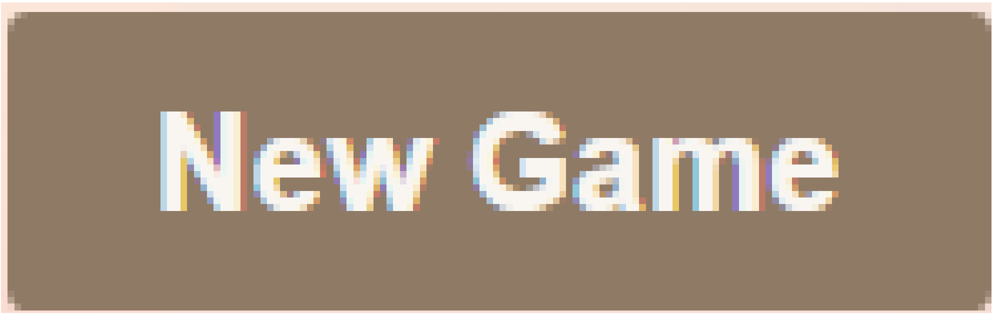
NO

Animation

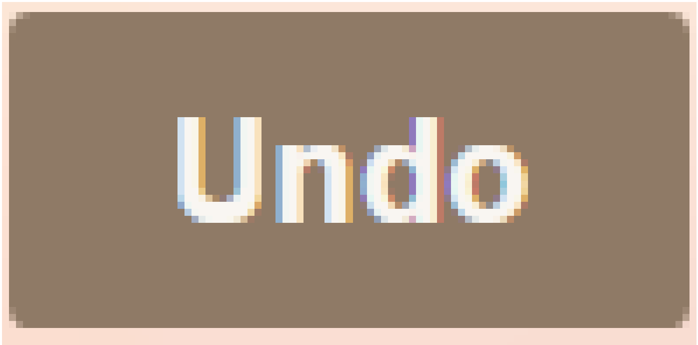
- **Animations:** "Move-up" and "fade-in" provide visual feedback for user actions and messages.
- **Tile Effects:** "Pop" effect highlights new and merged number tiles.

User Interaction

- **Button Styling:** Bold backgrounds and prominent text colors for "Restart" and "Undo" buttons.
- **Attention-Grabbing** Elements: Links (a) and important text (strong.important) styled to attract attention.



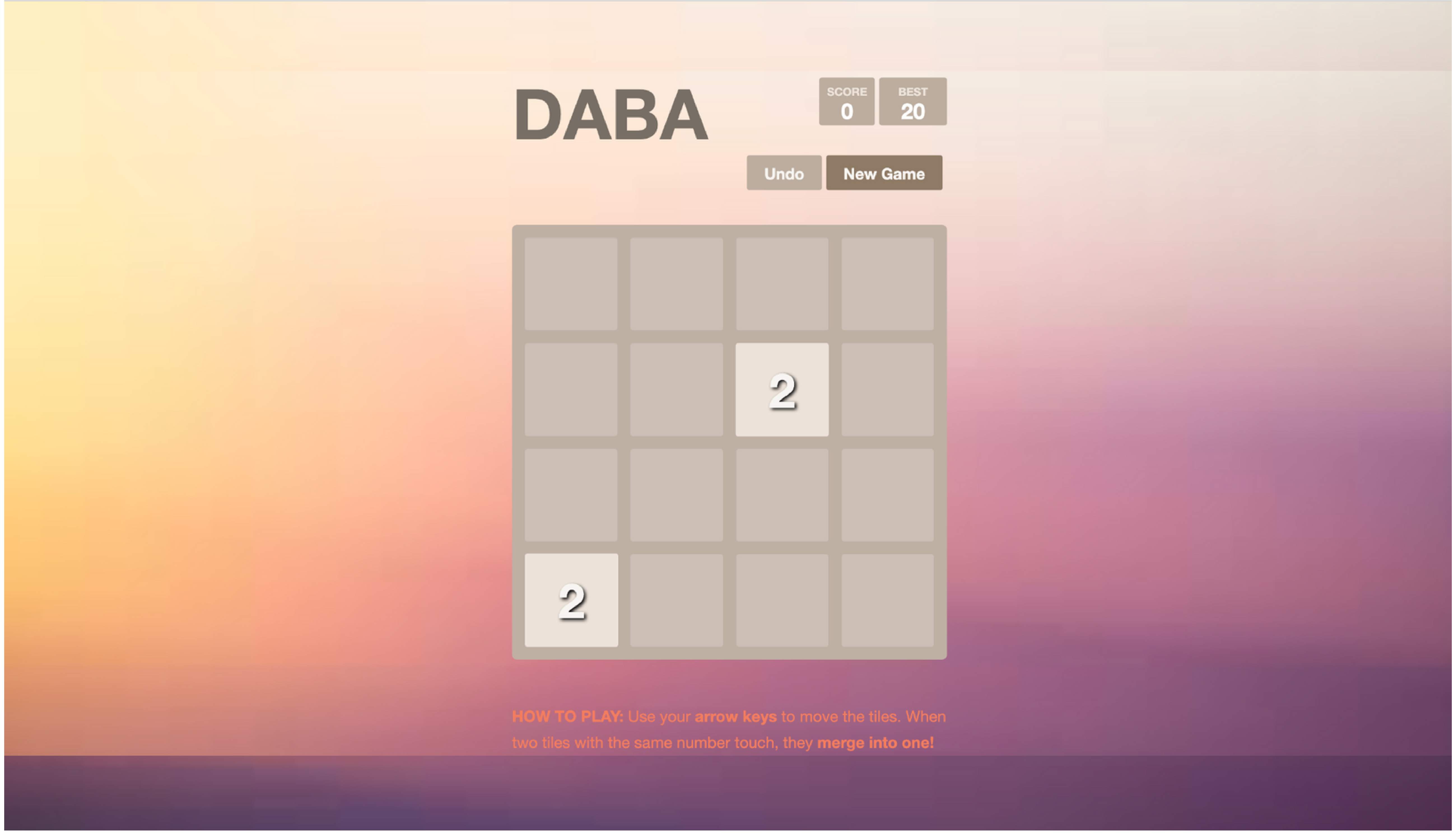
“New Game” Button



“Undo” Button

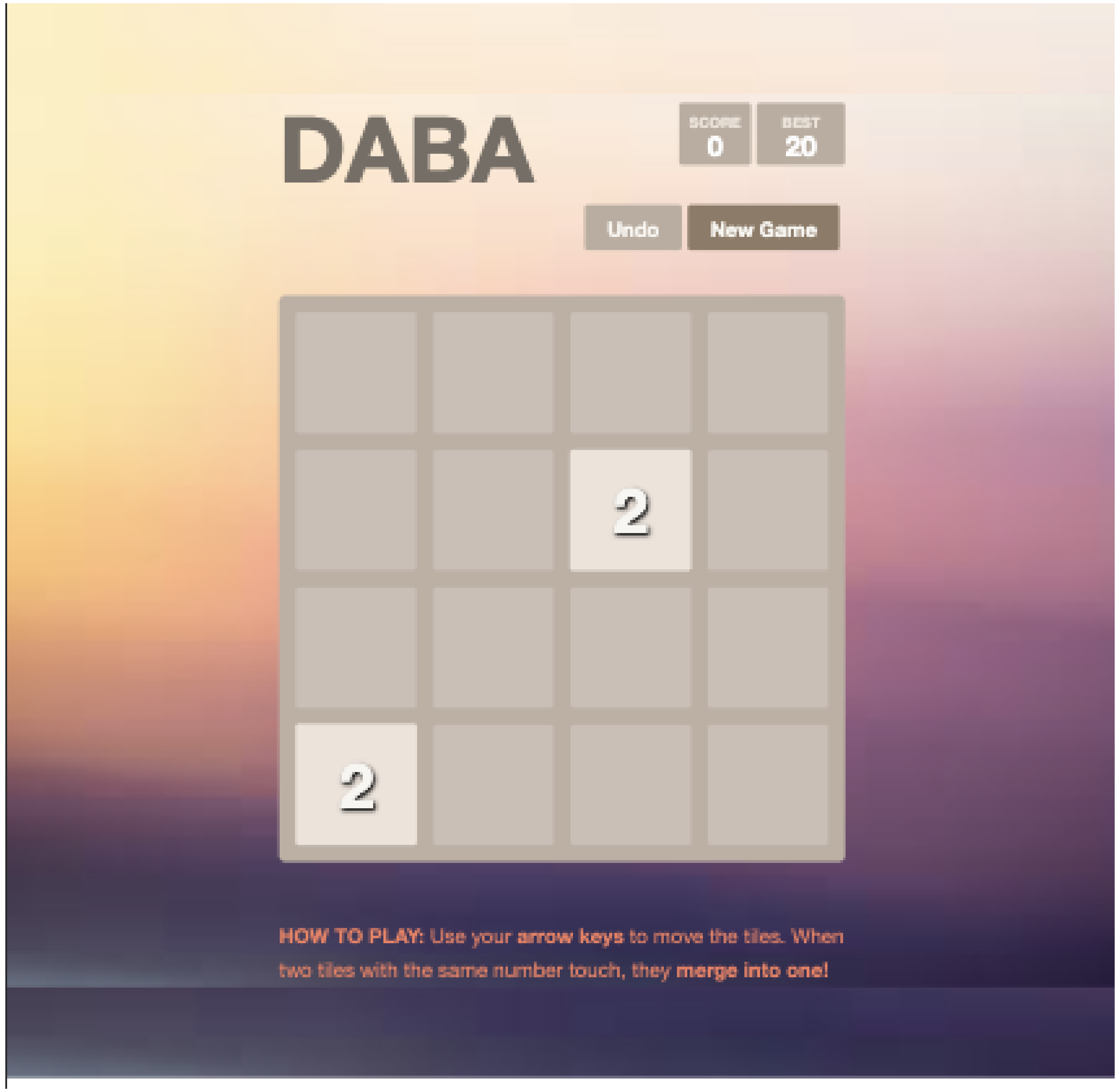
Mobile responsive

- **Responsive Design:** CSS media queries adapt game appearance for smaller screens, ensuring comfortable gameplay on various devices.



HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

User interface in Desktop



HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

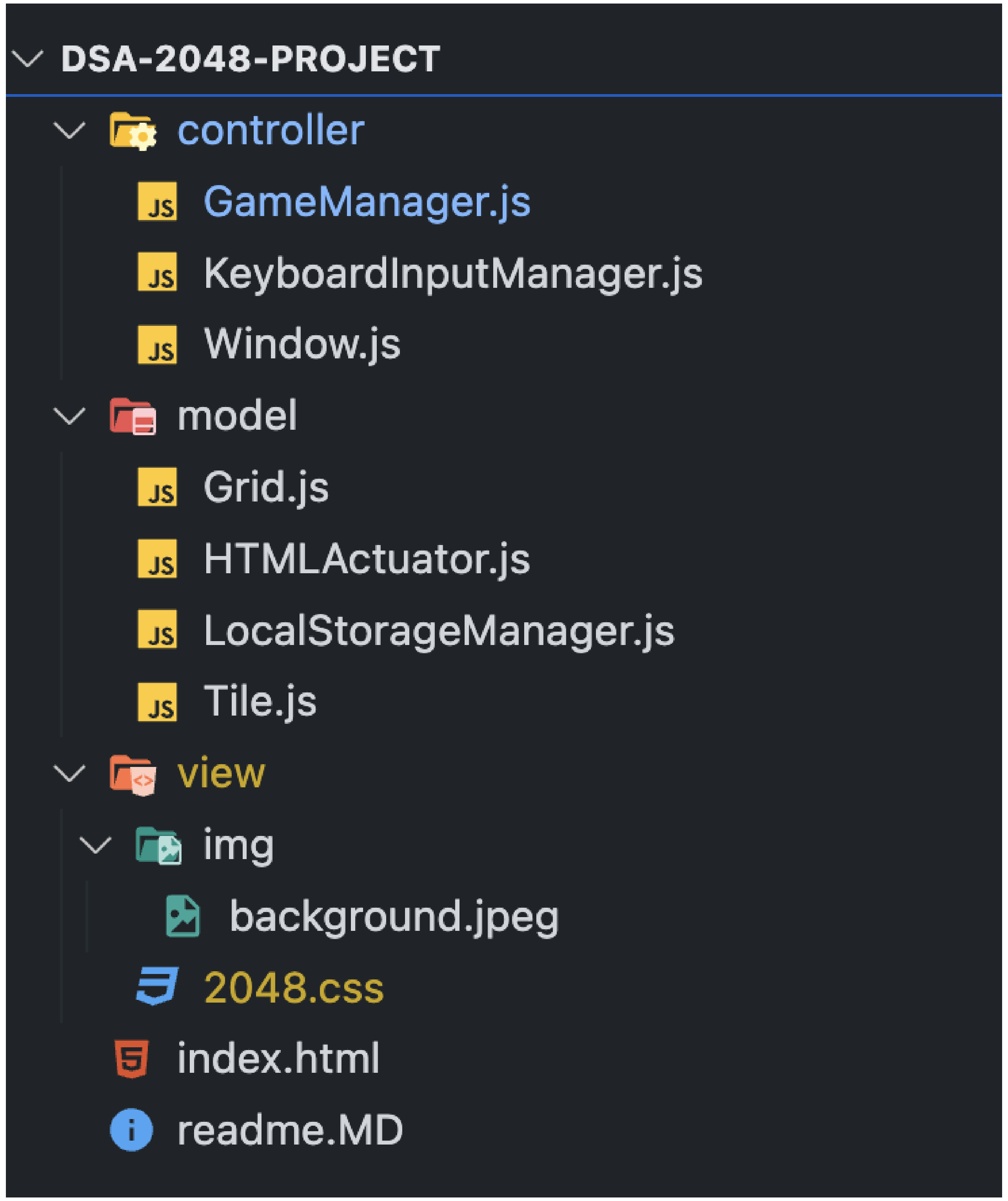
User interface in Mobile Phone

Design Pattern

YES NO

MVC Model in DABA 2048 Game

- **Model:**
  - **Grid.js:** Manages game grid and its elements.
  - **Tile.js:** Handles state of number tiles.
  - **LocalStorageManager.js:** Stores and retrieves data from local storage
- **View:**
  - **index.html:** UI design
  - **2048.css:** Styling
- **Controllers:**
  - **GameManager.js:** Main game handling
  - **KeyboardInputManager.js:** User input
  - **Windows.js**

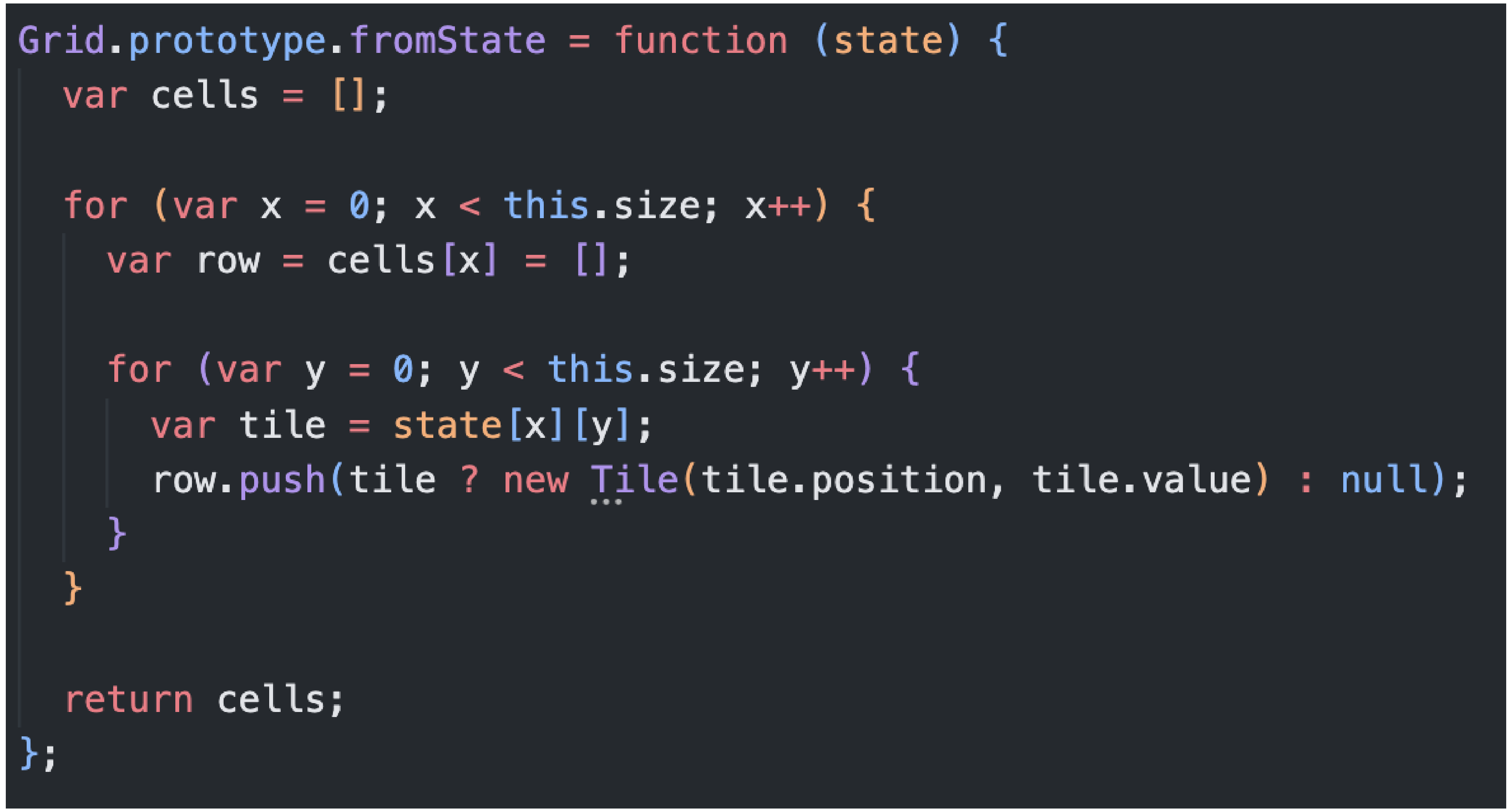


```
DSA-2048-PROJECT
  controller
    GameManager.js
    KeyboardInputManager.js
    Window.js
  model
    Grid.js
    HTMLActuator.js
    LocalStorageManager.js
    Tile.js
  view
    img
      background.jpeg
    2048.css
  index.html
  readme.MD
```

Using Data Structure and Algorithms

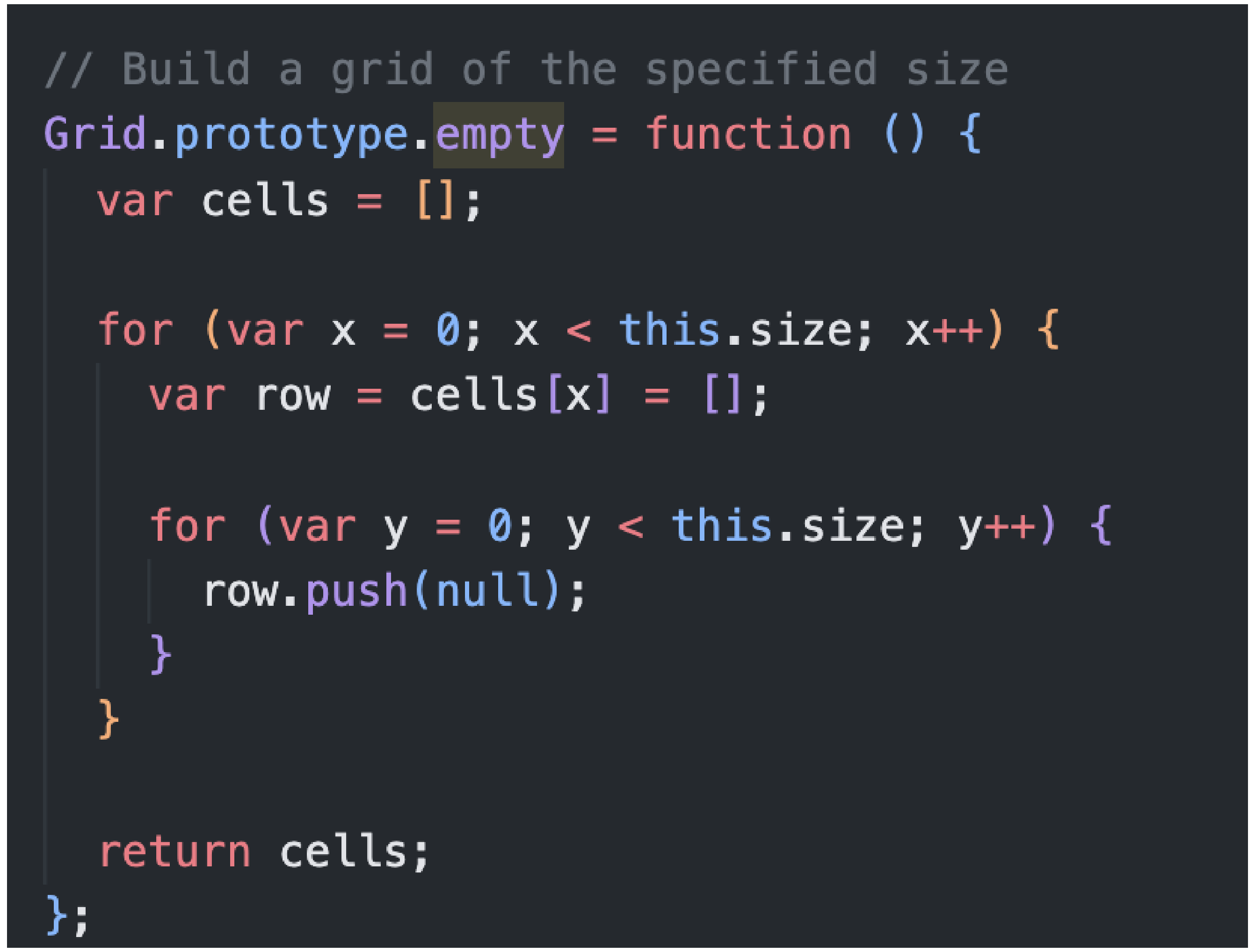
YES NO

Arrays – Grid management



```
Grid.prototype.fromState = function (state) {
  var cells = [];

  for (var x = 0; x < this.size; x++) {
    var row = cells[x] = [];

    for (var y = 0; y < this.size; y++) {
      var tile = state[x][y];
      row.push(tile ? new Tile(tile.position, tile.value) : null);
    }
  }

  return cells;
};
```

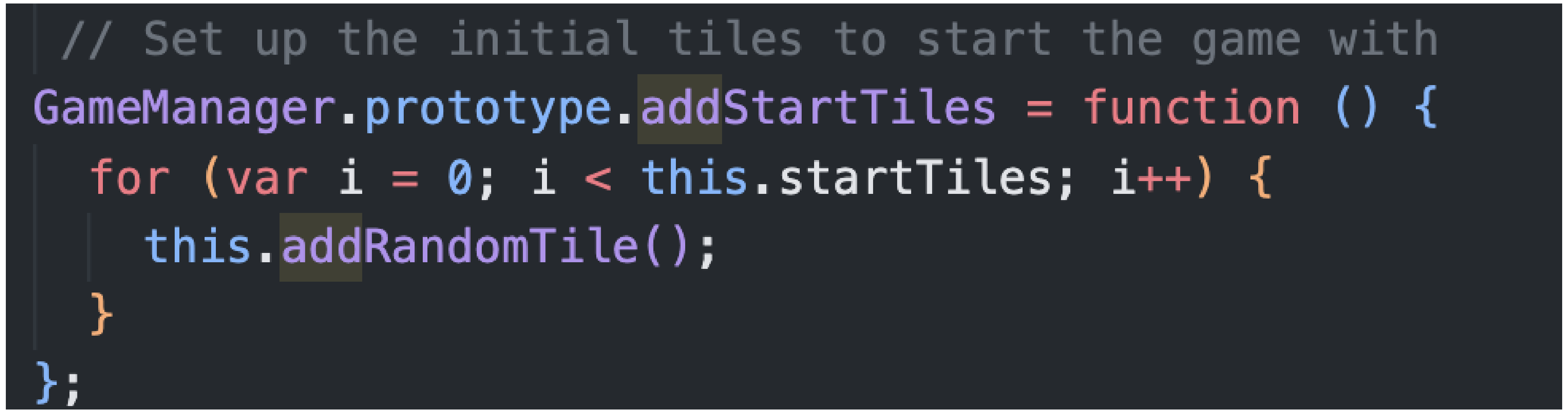
Grid.prototype.fromState method.



```
// Build a grid of the specified size
Grid.prototype.empty = function () {
  var cells = [];

  for (var x = 0; x < this.size; x++) {
    var row = cells[x] = [];

    for (var y = 0; y < this.size; y++) {
      row.push(null);
    }
  }

  return cells;
};
```

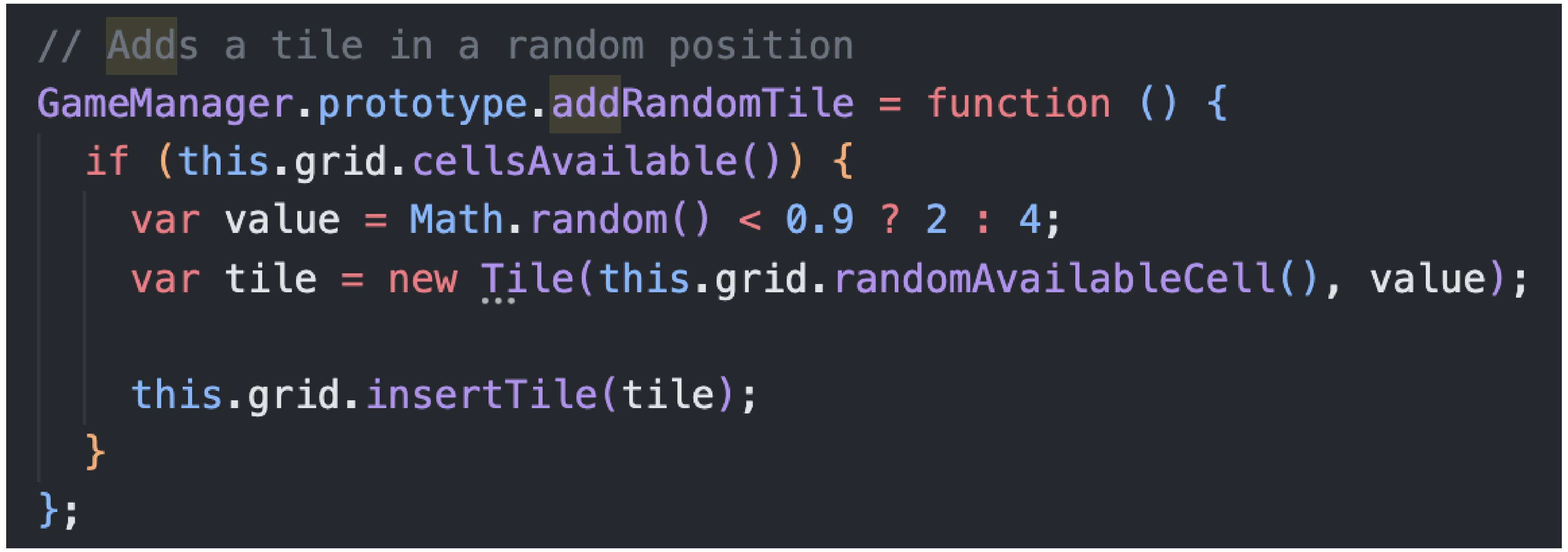
Grid.prototype.empty method.

Arrays – Tile managements



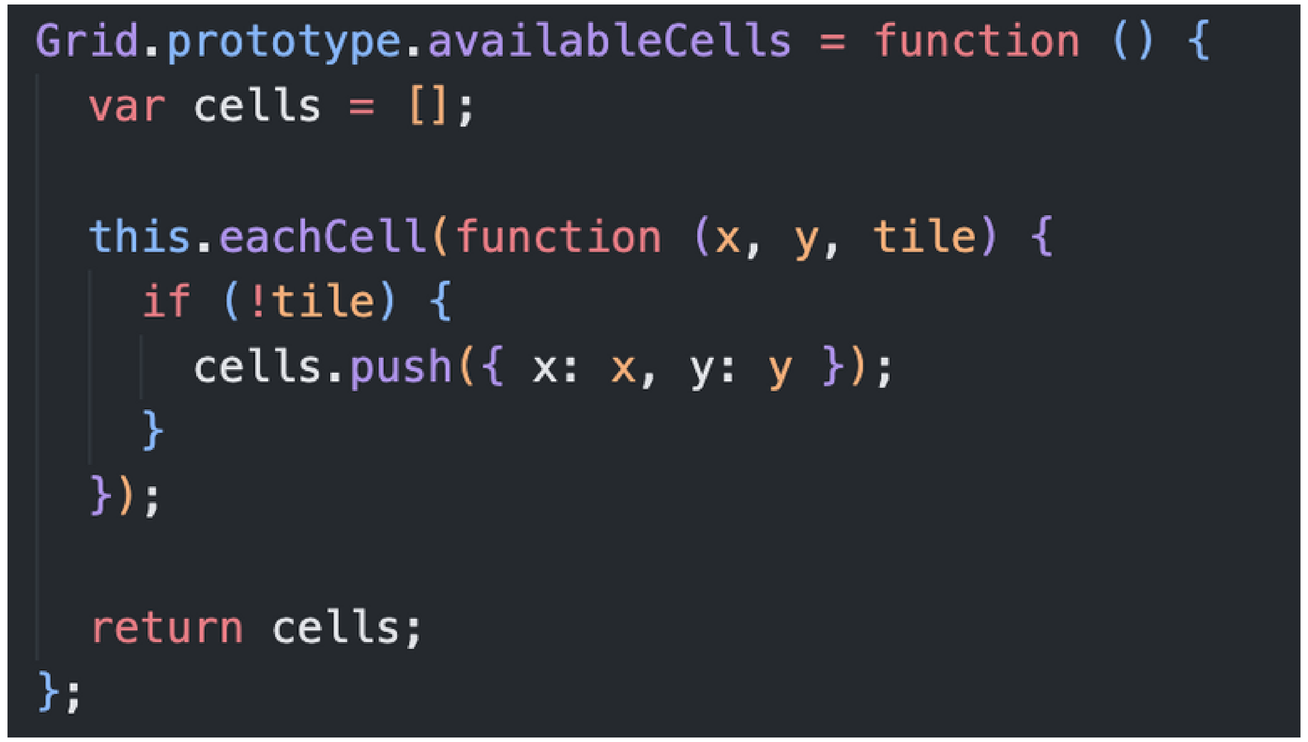
```
// Set up the initial tiles to start the game with
GameManager.prototype.addStartTiles = function () {
  for (var i = 0; i < this.startTiles; i++) {
    this.addRandomTile();
  }
};
```

GameManager.prototype.addStartTiles method.



```
// Adds a tile in a random position
GameManager.prototype.addRandomTile = function () {
  if (this.grid.cellsAvailable()) {
    var value = Math.random() < 0.9 ? 2 : 4;
    var tile = new Tile(this.grid.randomAvailableCell(), value);

    this.grid.insertTile(tile);
  }
};
```

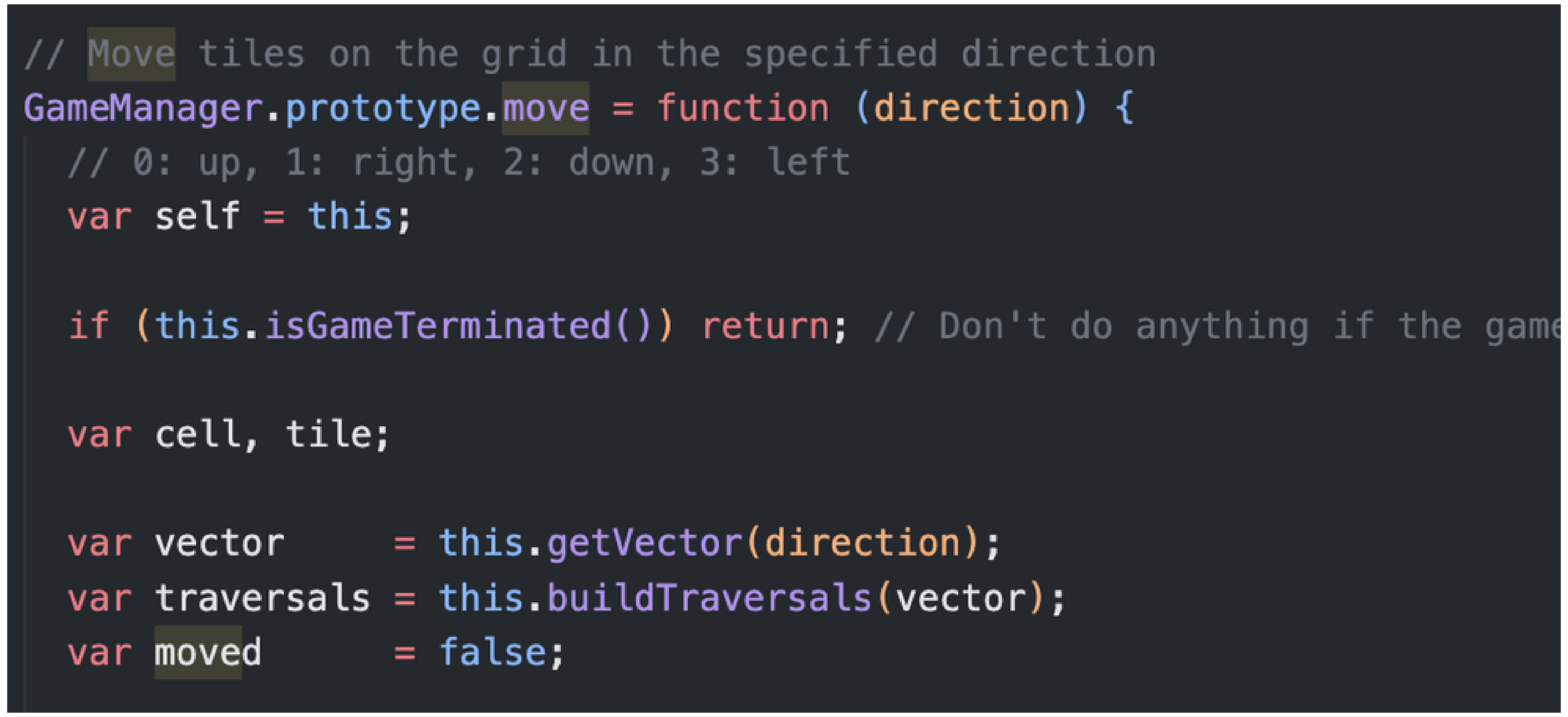
GameManager.prototype.addRandomTiles method.



```
Grid.prototype.availableCells = function () {
  var cells = [];

  this.eachCell(function (x, y, tile) {
    if (!tile) {
      cells.push({ x: x, y: y });
    }
  });

  return cells;
};
```

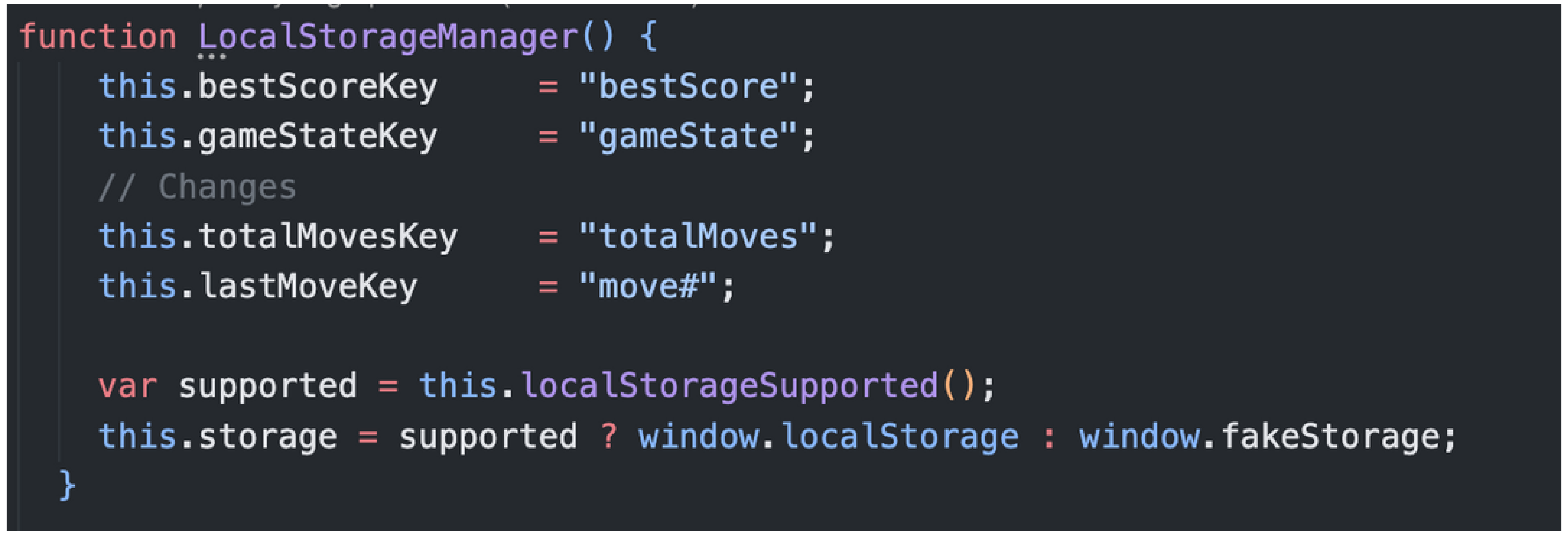
Grid.prototype.availableCells method.

Arrays – Tile managements



```
// Move tiles on the grid in the specified direction
GameManager.prototype.move = function (direction) {
  // 0: up, 1: right, 2: down, 3: left
  var self = this;

  if (this.isGameTerminated()) return; // Don't do anything if the game

  var cell, tile;

  var vector     = this.getVector(direction);
  var traversals = this.buildTraversals(vector);
  var moved      = false;
```

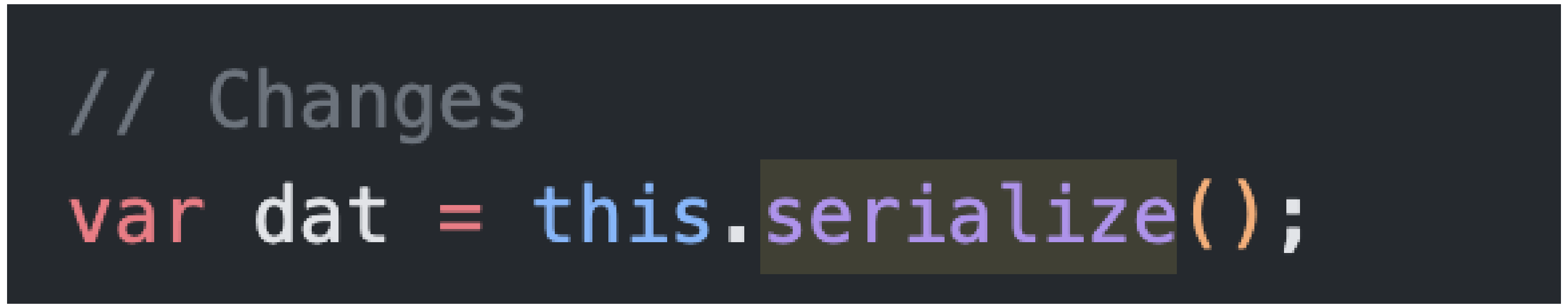
GameManager.prototype.move method.



```
function LocalStorageManager() {
    this.bestScoreKey     = "bestScore";
    this.gameStateKey     = "gameState";
    // Changes
    this.totalMovesKey    = "totalMoves";
    this.lastMoveKey      = "move#";

    var supported = this.localStorageSupported();
    this.storage = supported ? window.localStorage : window.fakeStorage;
  }
```

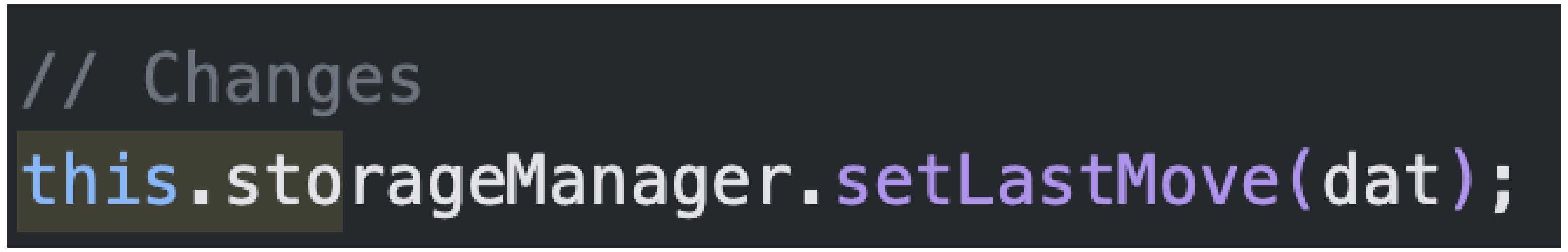
LocalStorageManager function.

Stack



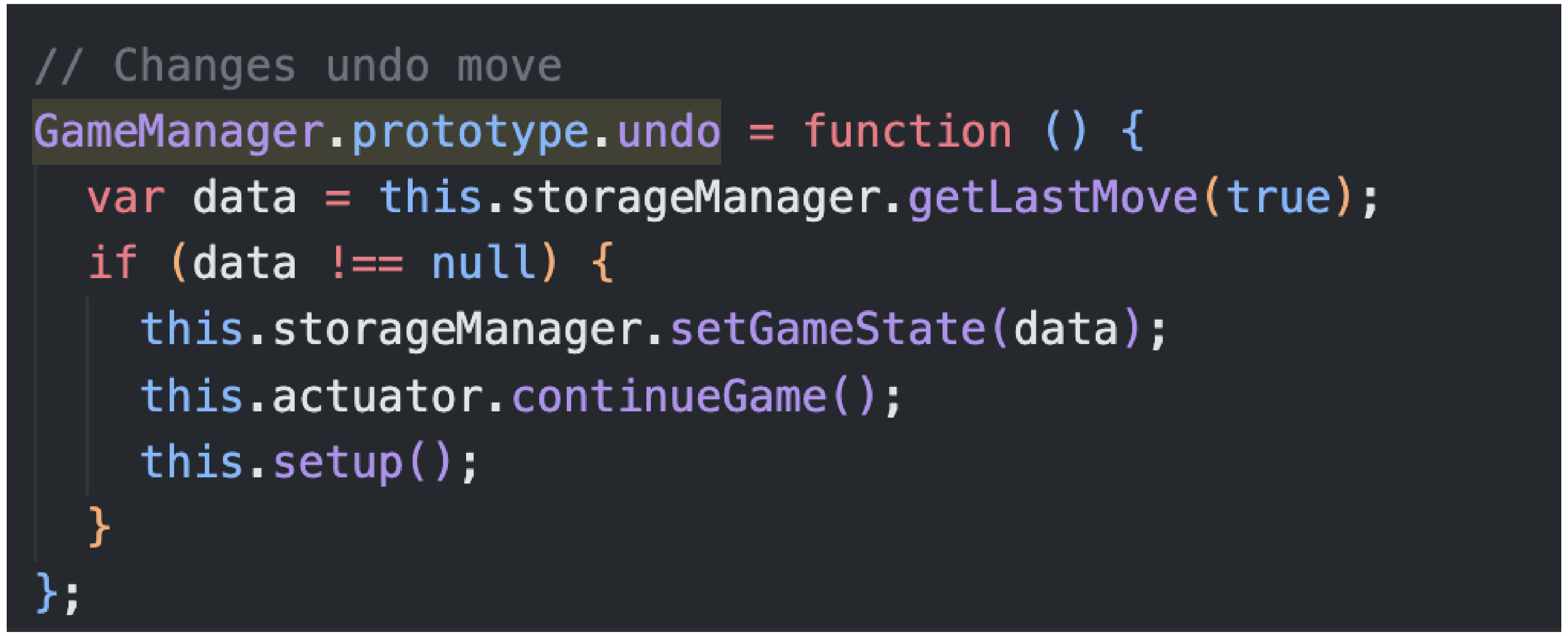
```
// Changes
var dat = this.serialize();
```

Save Status Before Moving.



```
// Changes
this.storageManager.setLastMove(dat);
```

Check If Any Move Is Performed.



```
// Changes undo move
GameManager.prototype.undo = function () {
  var data = this.storageManager.getLastMove(true);
  if (data !== null) {
    this.storageManager.setGameState(data);
    this.actuator.continueGame();
    this.setup();
  }
};
```

Undo move.

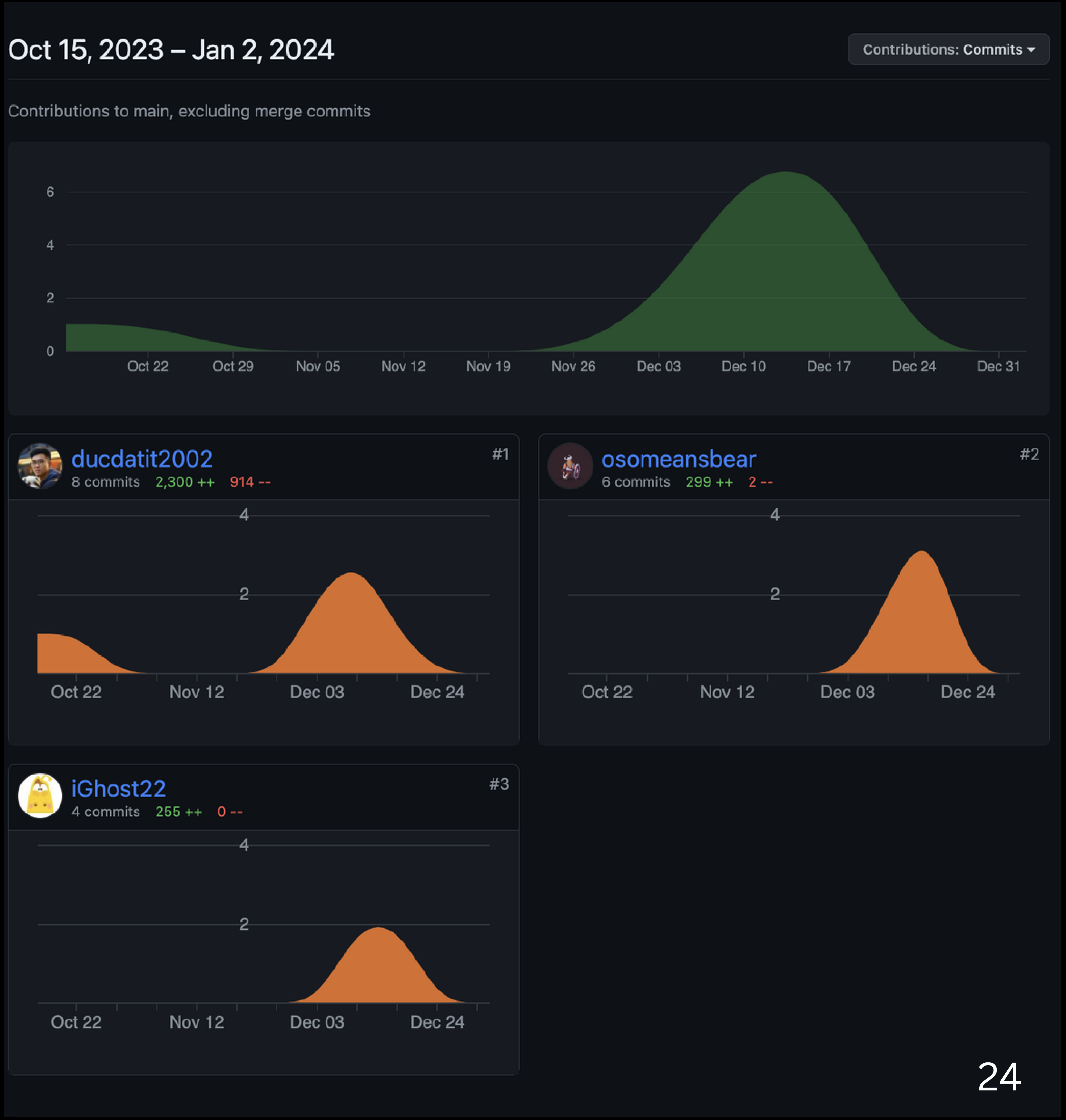
Git Commit History

Group Members

Phạm Đức Đạt – ITITIU20184

Phạm Vũ Bão – ITITWE20026

Huỳnh Lam Đạt – ITITIU20364



Oct 15, 2023 – Jan 2, 2024

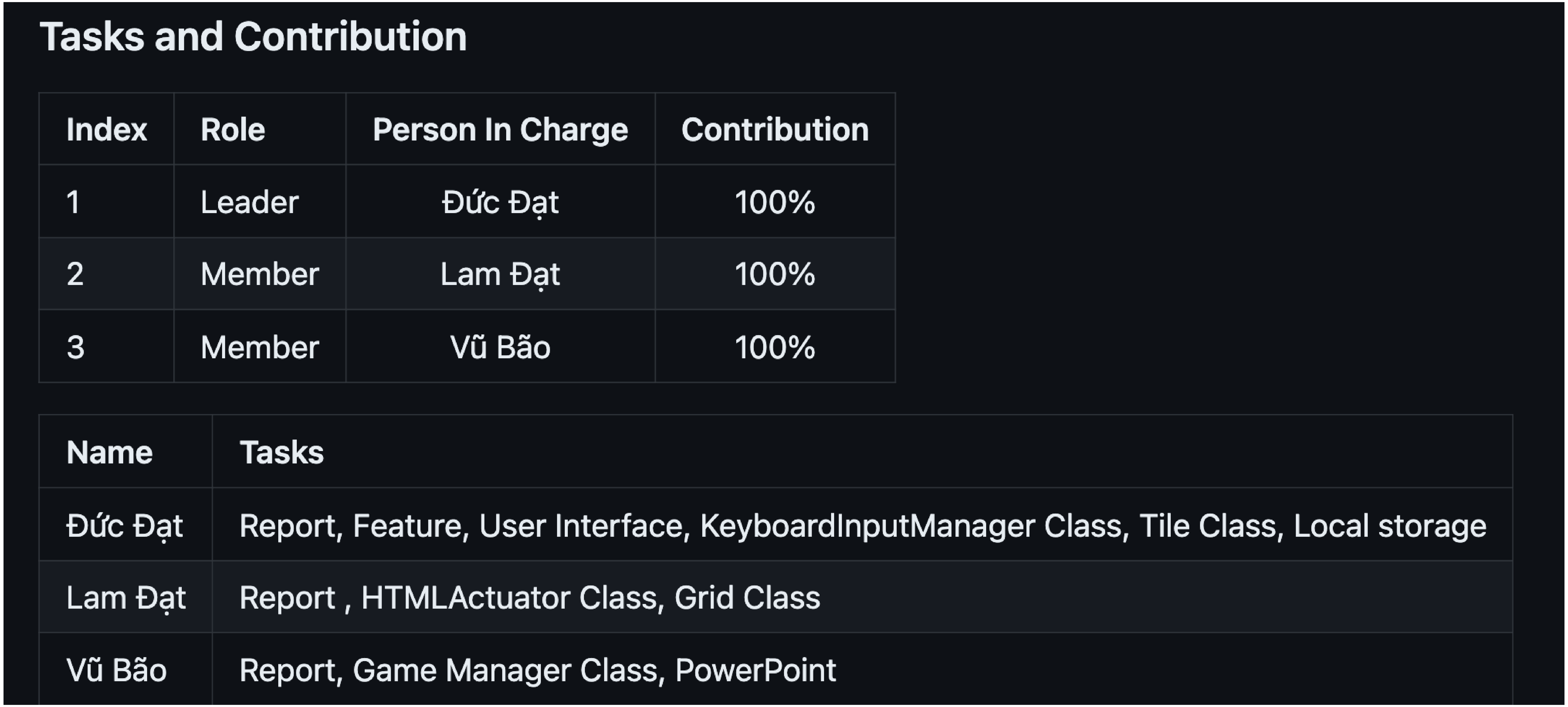
Contributions to main, excluding merge commits

ducdatit2002 #1
8 commits 2,300 ++ 914 --

osomeansbear #2
6 commits 299 ++ 2 --

iGhost22 #3
4 commits 255 ++ 0 --

Tasks & Contribution



Tasks and Contribution

Index	Role	Person In Charge	Contribution
1	Leader	Đức Đạt	100%
2	Member	Lam Đạt	100%
3	Member	Vũ Bão	100%

Name	Tasks
Đức Đạt	Report, Feature, User Interface, KeyboardInputManager Class, Tile Class, Local storage
Lam Đạt	Report , HTMLActuator Class, Grid Class
Vũ Bão	Report, Game Manager Class, PowerPoint

Conclusion

YES

NO

Conclusion

Respect for MVC Architecture

Improved user experience

Responsive design

Advanced Use of CSS and JavaScript

Effective data management

Git-based collaborative development

Potential for future expansion

DEMO

Thank you!

Do you have any question ?

YES NO